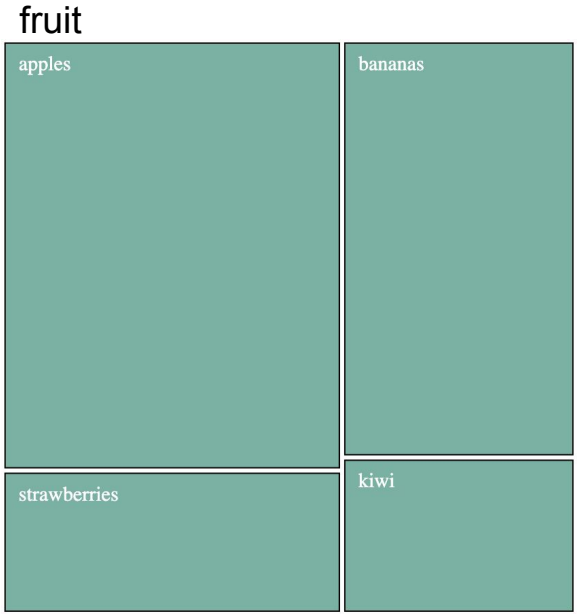
Treemaps

James Pretti

A Basic Treemap



(basic treemap of fruit)

name	parent	value
fruit		
apples	fruit	36
strawberries	fruit	12
bananas	fruit	24
kiwi	fruit	9

(csv file)

Name:
Parent = fruit
Child = a,s,b,k

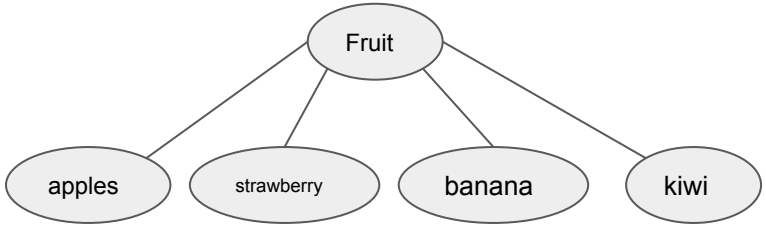
Parent:
Determines the parent, in this case it is “fruit”

Value:
Used to determine the size and location of each rectangle

How to make a basic treemap from a csv file

d3.stratify() : is used to create a hierarchy out of column data from a csv file

```
var root = d3.stratify()
  // Name of the child = column name in csv file
  .id(function(d) { return d.name; })
  // Name of the parent = column parent in csv file
  .parentId(function(d) { return d.parent; })
  (data);
```



root.sum : is used to get the value from the value column of the csv file and use it to determine the size of the treemap and value for each node

```
root.sum(function(d){return +d.value})
```

d3.treemap() : computes the position of each element in the hierarchy. The coordinates can then be added to the root object that was stratified earlier.

```
d3.treemap()
  .size([width, height])
  .padding(8)
  (root);
```

Rectangle Creation : in the following code, x0, x1, y0, and y1 are based on the value column from the csv. They represent the top left and bottom right corners of each rectangle. The proportion of the space that a rectangle covers inside the parent is based on the values from the value column in the csv

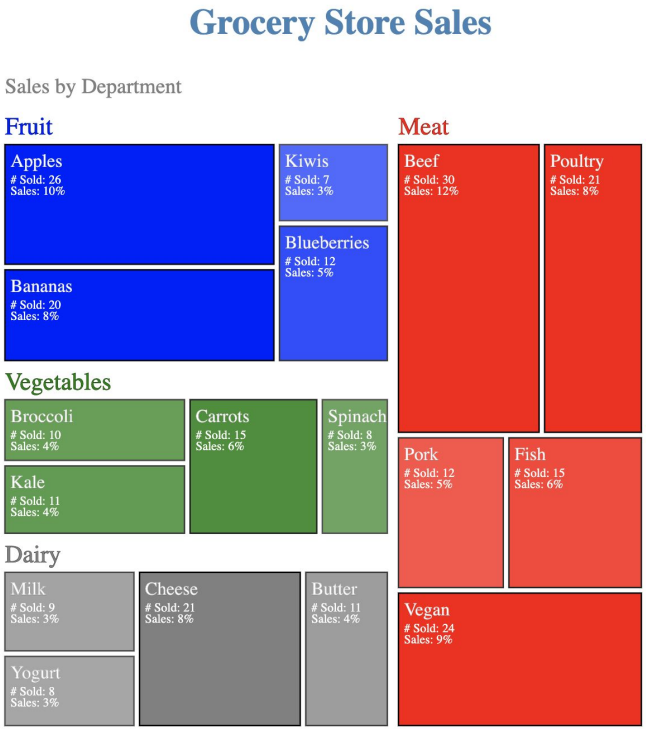
```
svg.selectAll("rect")
  // grabing data for all the children aka rect's
  .data(root.leaves())
  .enter()
  .append("rect")
    //this is where the location of each rect is set
    .attr('x', function(d){return d.x0;})
    .attr('y', function(d){return d.y0;})
    //this is where the size of each rect is set
    //depends on the value
    .attr('width', function(d){return d.x1 - d.x0; })
    .attr('height', function(d){return d.y1 - d.y0; })
    //some additional styling in css file
```

Basic is boring...

1) Recommended that more advanced treemaps use a json file rather than a csv because the data can be passed without the need for “reshaping” with stratify. This reducing the chart loading time.

2) There are many advanced uses with treemaps. For example, a treemap can house children’s children, essentially creating treemaps within treemaps.

Enhanced Treemap with json




```
{"children":
  [{"name":"Fruit","children":
    [{"name":"Apples","group":"A","value":26,"colname":"item"},
    {"name":"Bananas","group":"A","value":20,"colname":"item"},
    {"name":"Kiwis","group":"C","value":7,"colname":"item"},
    {"name":"Blueberries","group":"C","value":12,"colname":"item"}],
  "colname":"department"},

  {"name":"Vegetables","children":
    [{"name":"Broccoli","group":"C","value":10,"colname":"item"},
    {"name":"Kale","group":"A","value":11,"colname":"item"},
    {"name":"Carrots","group":"B","value":15,"colname":"item"},
    {"name":"Spinach","group":"B","value":8,"colname":"item"}],
  "colname":"department"},

  {"name":"Dairy","children":
    [{"name":"Milk","group":"C","value":9,"colname":"item"},
    {"name":"Yogurt","group":"A","value":8,"colname":"item"},
    {"name":"Cheese","group":"B","value":21,"colname":"item"},
    {"name":"Butter","group":"B","value":11,"colname":"item"}],
   "colname":"department"},

  {"name":"Meat","children":
    [{"name":"Beef","group":"B","value":30,"colname":"item"},
    {"name":"Poultry","group":"A","value":21,"colname":"item"},
    {"name":"Pork","group":"A","value":12,"colname":"item"},
    {"name":"Fish","group":"D","value":15,"colname":"item"},
    {"name":"Vegan","group":"D","value":24,"colname":"item"}],
  "colname":"department"}],

"name":"Store"}
```

New code

d3.hierarchy : this can be used with a json file instead of having to go through the process of reshaping each column into a hierarchy format like we did with the csv file using stratify.

```
var root = d3.hierarchy(data).sum(function(d){ return d.value})
// Here the size of each leaf is given in the 'value' field in
input data
```

Color : the color of each group of children can be adjusted individually.

```
var color = d3.scaleOrdinal()
  .domain(["Fruit", "Vegetables", "Dairy", "Meat"])
  .range([ "blue", "green", "grey", "red"])
```